Construction de trames sous contraintes temps réel

Ricardo Santos Marques, Nicolas Navet, Françoise Simonot-Lion

4 mars 2003

LORIA - TRIO

2, Avenue de la forêt de Haye

54516 Vandoeuvre-lès-Nancy

{rsantos,nnavet,simonot}@ensem.inpl-nancy.fr

Tel : 03.83.59.55.77 - Fax : 03.83.59.56.62

Correspondant : Ricardo Santos Marques

Résumé

La messagerie d'une application embarquée dans un véhicule doit respecter deux contraintes : elle doit d'abord être faisable d'un point de vue ordonnançabilité et elle doit minimiser, dans la mesure du possible, la consommation de bande passante. Ce dernier point est important pour permettre l'emploi de composants électroniques à faible coût et favoriser une conception incrémentale des applications. Dans cette étude, nous proposons deux heuristiques pour la génération automatique de messageries faisables qui minimisent l'utilisation de la bande passante. Les stratégies proposées sont complémentaires, la première est utilisable sur des problèmes de grande taille (dans le contexte du multiplexage véhicule), la seconde, plus efficace sur nos tests, ne peut être appliquée que sur des problèmes de taille restreinte (moins de 12 signaux par ECU). Nos propositions se sont révélées efficaces par rapport à d'autres stratégies a priori envisageables.

Mots clés : multiplexage véhicule, middleware, ordonnancement, messagerie, heuristique.

1 Introduction

Le contexte de ce papier est la conception de systèmes distribués embarqués dans l'automobile. Plus précisément, nous proposons une méthode pour construire automatiquement la messagerie hors ligne dans le cas où le réseau de communication utilisé est de type " bus à priorités ". Ces travaux trouvent leur application dans le cadre de la configuration des fonctions de communication d'un middleware embarqué qui masque les services du système de communication aux processus émetteur et récepteur des données, appelées ici, signaux, échangées sur le réseau. Cette problématique est primordiale dans le domaine des systèmes embarqués dans l'automobile. D'une part, les calculateurs connectés sur le réseau sont hétérogènes et, d'autre part, en raison du processus de développement spécifique à ce domaine applicatif, un calculateur muni de son logiciel (ECU - Electronic Control Unit) peut être développé par un équipementier puis intégré par un constructeur automobile pour lequel il est vu comme une boîte noire dont seuls sont connus les signaux produits et consommés. Dès lors, l'interopérabilité des différents ECUs au sein d'un même système doit être garantie. Elle ne s'exprime pas uniquement sur les valeurs des signaux mais aussi sur les propriétés temporelles attachées à chaque signal.

Les applications embarquées sont soumises à des contraintes de temps ; aussi, à chaque signal émis par un ECU est généralement associée une propriété sur sa durée de vie, que nous assimilons ici à une contrainte d'échéance sur sa date de réception par chaque ECU consommateur. Dans ce papier, nous considérons que, pour chaque signal, cette dernière propriété se réduit à la plus stricte des contraintes d'échéance appliquée à ce signal. Concevoir une messagerie consiste alors à résoudre le problème suivant : connaissant l'ensemble des ECUs et, pour chacun d'eux, l'ensemble des signaux produits et destinés à être émis sur un réseau, leur taille et leur période de production, il s'agit de construire l'ensemble des trames, la séquence de signaux composant chacune d'elle ainsi que sa période d'émission. Dans le cas particulier d'un bus à priorité, il faut en outre définir la priorité de chaque trame qui aura naturellement une influence sur la capacité des signaux de la trame à respecter leur contrainte d'échéance.

La messagerie de chaque ECU doit être construite de façon à minimiser la consommation de bande passante. La première raison est de préserver la possibilité d'adjonction de nouvelles fonctionnalités (sous la forme d'un ou plusieurs ECUs) qui ajouteraient de nouveaux signaux à la messagerie. Une telle conception incrémentale est une pratique courante dans l'industrie automobile. La seconde raison de minimiser la consommation de bande passante est de permettre l'utilisation de processeurs de moindre puissance, donc moins coûteux.

Le problème à résoudre est donc un problème d'optimisation de la messagerie (ensemble de trames, placement des signaux, ordonnancement des trames) sous une contrainte d'ordonnançabilité et avec comme critère d'optimisation la minimisation de la bande passante. Ce problème est NP-complet et ne peut se résoudre en utilisant une approche strictement exhaustive au delà d'un très petit nombre de signaux et / ou d'ECUs. La solution passe donc par la recherche d'heuristiques.

Une solution au problème de construction de messageries sous contraintes d'ordonnançabilité est proposée dans le même contexte applicatif par le middleware Volcano (cf. [9]) mais les algorithmes employés dans ce produit commercial ne sont pas publiés. Par ailleurs, dans [4] il est proposé des heuristiques pour construire une messagerie sur CAN qui minimise la bande passante mais sans rechercher explicitement une solution faisable ; par ailleurs, la solution proposée n'affecte pas de priorité aux messages construits. Enfin, dans le contexte de la gestion de production, ont été développées de multiples stratégies pour placer des éléments de tailles différentes dans des boîtes (cf. [7] pour un état de l'art). Ce problème est connu usuellement sous le terme de " bin-packing ".

Dans ce papier, nous proposons deux algorithmes heuristiques : le premier, inspiré de stratégies de bin-packing, a une complexité algorithmique qui lui permet d'être utilisé sur des problèmes de taille réaliste. La seconde heuristique, sans effectuer un parcours exhaustif, explore plus largement l'espace des solutions et n'est applicable que sur des problèmes de taille limitée mais avec une très bonne efficacité. Dans cette étude, les métriques de performance sont la consommation de bande passante et la capacité à trouver des solutions faisables du point de vue ordonnançabilité. Nos propositions seront évaluées par rapport à deux heuristiques efficaces pour des problèmes de bin-packing et une stratégie naïve consistant à placer un signal par trame.

La section 2 est consacrée à la présentation détaillée du problème, de sa complexité algorithmique, des hypothèses faites dans cette étude ainsi que de la technique de vérification de la faisabilité qui sera utilisée. Enfin dans les sections 3 et 4, nous présenterons les deux heuristiques proposées et évaluerons leurs performances.

2 Modèle et notations

Le problème est de constituer un ensemble de k trames $F = \{f_1, f_2, ..., f_k\}$ à partir d'un ensemble de n signaux $S = \{s_1, s_2, ..., s_n\}$ de telle façon à minimiser la consommation de bande passante tout en respectant la contrainte de

localité (un signal ne peut être émis que par la station qui le produit) et les contraintes d'échéances sur les dates de réception des signaux. Chaque signal s_i est caractérisé par :

- sa station émettrice N_i,
- sa période de production T_i sur la station émettrice. En pratique, les horloges des stations ne sont pas synchronisées et nous pouvons avoir des décalages initiaux quelconques sur les dates de première production de signaux situés sur des stations différentes. Par contre, nous considérons que sur une même station les processus de production des signaux sont synchronisés (ils débutent tous au même instant),
- sa taille en bits C_i avec l'hypothèse que C_i est toujours inférieure ou égale à la taille maximale des données d'une trame (pas de segmentation),
- son échéance relative notée D_i, c'est l'intervalle de temps maximum entre la mise à disposition du signal du coté producteur et sa réception par le ou les destinataires. Nous ferons l'hypothèse que l'échéance est égale à la période de production mais les stratégies proposées restent valables pour des échéances inférieures à la période.

Le réseau de communication considéré est un bus à priorités qui pourra être CAN, VAN ou J1850 qui sont des standards de fait dans les applications de multiplexage véhicule. Les applications numériques seront faites avec un réseau CAN à 500kbit/s (débit classique d'un réseau "moteur") où les trames auront un overhead de 64bits[1]. Chaque trame f_i résultant de processus de construction aura les caractéristiques suivantes :

- sa station émettrice N_i^*,
- sa période d'émission T_i^* qui est la période la plus petite des signaux qui la composent,
- sa taille en bits C_i^* qui est constituée d'une partie de donnée et d'un overhead indépendant de la taille des données,
- son échéance relative notée D_i^* qui est fonction des caractéristiques des signaux qui la composent,
- sa priorité P_i^* vis-à-vis du protocole de communication.

Il est habituel dans les réseaux de multiplexage que du trafic non-temps réel (et de priorité faible) comme des trames de diagnostic circulent en marge du trafic temps réel. Nous ferons ici l'hypothèse que ces trames ont une taille de 128 bits (8 octets de données plus 64 bits d'overhead) qui constituera le facteur de blocage pour toutes les trames de notre système (ie. l'intervalle de temps maximum

[1] La taille exacte de l'overhead d'une trame CAN n'est pas indépendante du contenu des données à cause du mécanisme de "bit-stuffing", un overhead de 64 bits par trame est une valeur raisonnable.

pendant lequel une trame prioritaire peut être retardée par une trame moins prioritaire).

2.1 Complexité du problème

Le problème de la construction de trames à partir d'un ensemble de signaux s'apparente à un problème de bin-packing et a été prouvé NP complet dans [4]. Néanmoins sur des problèmes de petite taille, une approche exhaustive pourrait être envisageable c'est pourquoi nous déterminons la complexité exacte du problème.

Regrouper un ensemble de n signaux en k trames non-vides revient à créer toutes les partitions de taille k de l'ensemble des signaux. La complexité de ce problème est connu, il s'agit du nombre de Stirling du second ordre (cf. [1] page 824) :

$$\frac{1}{k!}\sum_{i=0}^{k}(-1)^{(k-i)}\binom{k}{i}i^n.$$

Le nombre de trames k par station peut varier de 1 jusqu'à n où n est le nombre de signaux produits par la station. Le nombre de trames à envisager sur une station i qui comporte n signaux est donc :

$$\mathcal{S}_i = \sum_{k=1}^{n}\frac{1}{k!}\sum_{i=0}^{k}(-1)^{(k-i)}\binom{k}{i}i^n.$$

Ainsi trois signaux a, b et c induisent 5 combinaisons différentes : $[(a,b,c)]$, $[(a),(b,c)]$, $[(a,b),(c)]$, $[(a,c),(b)]$ et $[(a),(b),(c)]$.

Si l'on considère un ensemble de m stations, l'espace des solutions devient $\prod_{i=1..m} S_i$ où S_i est le nombre de trames possibles sur la station i. L'espace des solutions est rapidement prohibitivement grand dès que n et m croissent. Par exemple pour 10 signaux par stations ($S_i = 115975$) et 5 stations, une recherche exhaustive devrait parcourir environ $2 \cdot 10^{25}$ solutions. Notons, de plus, que à ce calcul s'ajoute la recherche des priorités des trames pour chaque solution envisagée. On voit que même sur des problèmes de taille modeste une approche exhaustive est absolument inenvisageable ce qui justifie pleinement l'utilisation d'heuristiques.

2.2 Allocation de priorités optimale

Sous la politique Fixed Priority Preemptive (FPP), le temps de réponse d'une tâche ne dépend que de l'ensemble des tâches plus prioritaires et non des

priorités relatives entre les tâches plus prioritaires puisque quoi qu'il arrive il faudra exécuter tout le travail plus prioritaire. En partant de cette observation, Audsley dans [2] a proposé un algorithme d'allocation de priorités en $O((n^2 + n)/2)$ qui est optimal dans le sens où, si une solution existe, alors elle sera nécessairement trouvée par cet algorithme. L'idée est de partir du niveau de priorité le plus faible (m) et de chercher une tâche qui soit faisable à ce niveau de priorité. Si il n'y a aucune tâche faisable au niveau de priorité m, alors l'ensemble des tâches est nécessairement non-faisable. La première tâche faisable au niveau m se voit affecter cette priorité, il est clair que cette tâche aurait été faisable à tous les autres niveaux de priorité. Une fois le niveau m attribué, l'algorithme cherche une tâche faisable au niveau $m - 1$ et ainsi de suite jusqu'à la priorité 1 qui est la plus forte priorité du système.

Dans le cas général cet algorithme n'est pas applicable à l'ordonnancement de messages sous Non-Preemptive Fixed Priority (NPFP) qui est la politique utilisée pour l'accès au médium sur un bus à priorités. En effet, le temps de réponse d'une trame dépend non seulement de la quantité de travail plus prioritaire mais également de l'ensemble des trames moins prioritaires à cause du facteur de blocage (le plus grand temps pendant lequel une trame peut être retardée par une trame moins prioritaire). Dans notre contexte particulier, le facteur de blocage est identique pour toutes les trames car nous considérons l'existence d'un trafic non-temps réel (trames de diagnostic) ce qui est le cas en pratique. Sans connaissance supplémentaire sur ce trafic, il nous faut considérer le facteur de blocage comme étant la taille de la plus grande trame compatible avec le protocole utilisé. Dans ce cas, les hypothèses sont remplies pour utiliser l'algorithme d'Audsley afin de juger de la faisabilité d'un ensemble de trames.

3 L'heuristique "Bandwith-Best-Fit decreasing"

Cette heuristique est ainsi appelée par analogie avec les dénominations utilisées dans le cadre des problèmes de bin-packing avec résolution hors-ligne *(Best-Fit decreasing - First-fit decreasing*, cf. [7]). Dans le domaine du bin-packing, l'objectif est de minimiser le nombre de boîtes (des trames ici), dans notre contexte l'objectif est de minimiser la quantité de bande passante sans tenir compte du nombre de trames. L'idée de l'heuristique est de placer un signal dans la trame telle que la quantité de bande passante consommée par ce signal soit minimale. L'heuristique construit une seule solution dont la faisabilité est testée avec l'algorithme d'Audsley. Si le test est négatif, l'heuristique applique des transformations sur les trames qui visent à déplacer de la charge (des

signaux) à des niveaux de priorités inférieures.

3.1 Description

La description algorithmique de l'heuristique "Bandwith-Best-Fit decreasing" est donnée ci-dessous :

1. sur chaque station, tri des signaux dans l'ordre décroissant de la consommation de bande passante.
2. placer le signal s_i dans une trame :
   (a) il existe au moins une trame dans la station qui puisse accepter s_i, c'est à dire une trame dont la taille des données incluant s_i est inférieure à la limite fixée par le protocole et dont la nouvelle échéance est strictement positive (cf. Appendice A). On insére s_i dans la trame f_k qui minimise la consommation de bande passante calculée avec s_i puis on ajuste les caractéristiques temporelles de la trame $T_k^* = \min(T_k^*, T_i)$ et D_k^* selon l'algorithme donné en Appendice A.
   (b) il n'existe aucune trame qui puisse accepter s_i : créer une nouvelle trame de caractéristiques temporelles $T_k^* = T_i$ et $D_k^* = D_i$.
3. tant qu'il reste un signal non-encore placé retour à l'étape 2.
4. test de la faisabilité de la messagerie avec l'algorithme d'Audsley :
   (a) la configuration n'est pas faisable :
      i. on construit $\hat{F}$ l'ensemble des trames pour lesquelles aucune priorité n'a été trouvée.
         A. il reste une trame de $\hat{F}$ avec au moins deux signaux : on choisit la trame dont l'échéance a été la moins dépassée au premier niveau de priorité non-pourvu et qui possède au moins deux signaux. Cette trame est notée $\hat{f}$.
         B. toutes les trames ont été entièrement décomposées : ECHEC.
      ii. on retire de l'ensemble des signaux de $\hat{f}$, celui, dont l'échéance est la plus petite, que l'on place dans une nouvelle trame. On met a jour les caractéristiques de $\hat{f}$ (période, échéance et taille).
      iii. retour à l'étape 4 (l'algorithme s'arrête lorsque toutes les trames non-faisables ont été complètement décomposées).
   (b) la configuration est faisable : SUCCES.

Cette heuristique est constituée de deux parties disjointes. La première partie (étapes 1 à 3) vise à construire une solution qui minimise la consommation de bande passante. Ces étapes sont inspirées de l'heuristique très compétitive "Best-Fit decreasing" du domaine du bin-packing avec comme critère de choix non plus la taille des objets, mais la consommation de bande passante. La seconde partie (étape 4) se préoccupe de la faisabilité de la solution proposée. En cas d'insuccès au test d'Audley, l'heuristique essaye de relaxer les contraintes des trames en isolant les signaux les plus exigeants. La trame choisie initialement pour la décomposition est celle qui dépasse le moins son échéance au premier niveau de priorité non-pourvu et donc qui a le plus de chance de respecter son échéance si elle contient moins de données. Pour déterminer quelle est cette trame, il est nécessaire d'effectuer un calcul de temps de réponse pour chacune des trames pour lesquelles aucune priorité n'a été encore trouvée (en leur affectant le premier niveau de priorité non pourvu par l'algorithme d'Audsley, les priorités supérieures sont inchangées et les priorités inférieures sont indifférentes).

L'étape 1 de l'heuristique est un tri donc sa complexité algorithmique est $n \log(n)$ (où n est le nombre de signaux de la station), les étapes 2 et 3 sont en $O((n^2 + n)/2)$ dans le pire cas. Au cours de l'étape 4, il y a au plus n calculs de temps de réponse (aucun niveau de priorité pourvu) et n appels à l'algorithme d'Audsley. Sur nos expérimentations menées sur des configurations de taille réaliste (une centaine de signaux répartis sur une dizaine de stations), cette complexité n'a pas posée en pratique de problème de temps de calcul.

3.2 Evaluation de performances

Les performances de l'heuristique "Bandwith-Best-Fit decreasing" (BBFd) sont évaluées vis-à-vis des deux métriques de performance importantes dans notre contexte qui sont la faisabilité du système et la consommation de bande passante. Les stratégies en concurrence sont :

- "un signal par trame" (1SpT) : chaque trame est constituée d'un seul signal,
- First-Fit decreasing (FFd) : les signaux sont triés dans l'ordre décroissant par taille, ils sont ensuite insérés dans la première trame qui peut les contenir,
- Best-Fit decreasing (BFd) : les signaux sont triés dans l'ordre décroissant par taille, ils sont insérés dans la trame qui aura le moins de place disponible après insertion.

Les heuristiques ont été implantées en C++ et la faisabilité de chaque solution envisagée est testée par le logiciel *rts* (écrit par Jörn Migge, cf. [3]) qui implante

l'algorithme d'Audsley (cf. paragraphe 2.2) et le calcul de bornes sur les temps de réponse pour le réseau CAN tel que proposé dans [11].

3.2.1 Consommation de bande passante

Nous ne considérons la charge que de configurations faisables sous toutes les stratégies car la mise en place de messageries non-faisables est exclue dans le contexte d'applications temps réel. La figure 1 représente la quantité de charge réseau moyenne sur 100 configurations faisables pour une charge utile variant de 15 à 35%, la charge utile étant uniquement constituée par les données. Les signaux sont générés aléatoirement avec une période variant de 5 à 100ms (tirage uniforme - pas de 5ms), une échéance égale à la période, une taille comprise entre 1 et 8 octets (tirage uniforme). Pour chaque test, le nombre de signaux généré est le nombre moyen nécessaire pour atteindre le niveau de charge utile désirée. Les signaux sont placés aléatoirement sur un ensemble de 10 stations. Le débit du réseau est de 500kbit/s et l'overhead est de 8 octets pour une trame de données qui peut contenir 8 octets de données (réseau CAN).

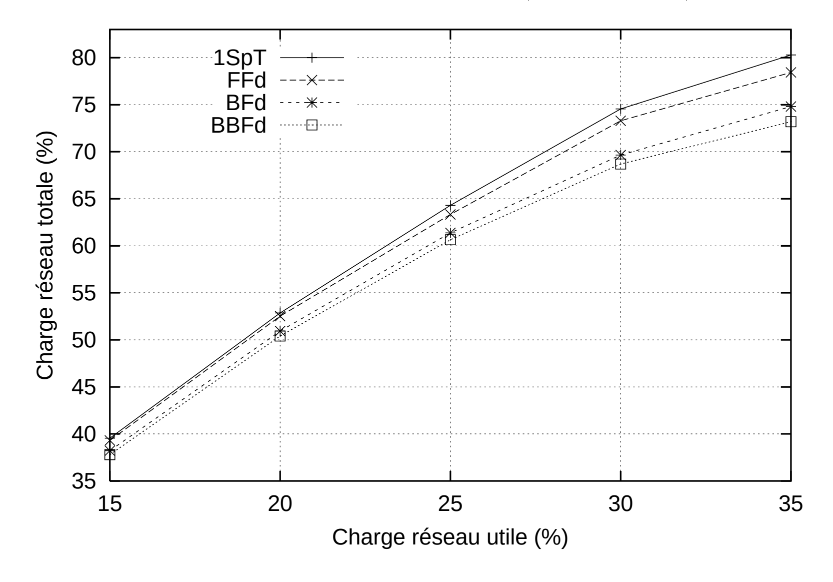


FIG. 1 – Charge réseau totale utilisée pour une charge utile variant de 15 à 35%. Résultats moyens sur 100 configurations faisables sous les 4 heuristiques en concurrence.

Comme on peut le voir sur la figure 1, les performances relatives des heuristiques conservent le même ordre lorsque la charge soumise augmente. BBFd donne toujours les meilleurs résultats avec un gain variant de 4,2% à 8,9% sur l'approche naïve 1SpT et de 1,1% à 2,2% sur BFd qui est l'heuristique la plus proche. Les gains en termes de consommation de bande passante sont réels à tous les niveaux de charge sans être considérables.

3.2.2 Faisabilité des configurations

Les conditions d'expérimentation sont celles décrites dans le paragraphe 3.2.1. Les tests portent sur 100 jeux d'essai générés aléatoirement. Le tableau 1 présente les résultats obtenus pour les 4 heuristiques en concurrence.

charge utile	15%	20%	25%	30%	35%
1SpT	100	100	100	97	36
FFd	100	100	100	98	54
BFd	100	100	100	99	76
BBFd	100	100	100	100	82

TAB. 1 – Nombre de configurations faisables sur 100 tests pour une charge utile variant de 15 à 35%.

Jusqu'à une charge utile de 25%, toutes les heuristiques conduisent à une solution faisable. Au delà, BBFd est toujours la meilleure et permet en particulier d'obtenir 6 configurations faisables supplémentaires au niveau de charge le plus élevé. Ceci nous suggère que la technique de relaxation des contraintes utilisée en cas de non-faisabilité est efficace (cf. étape 4 de l'algorithme au paragraphe 3.1).

4 L'heuristique "semi-exhaustive"

La dénomination "semi-exhaustive" a été choisie car la stratégie est un parcours exhaustif de l'espace des solutions dont on aurait coupé certaines branches jugées a priori peu prometteuses. Cet algorithme débute par la construction de l'ensemble de toutes les trames qu'il est possible de constituer sur chaque station. En pratique, la complexité de cette première étape, déterminée précisément dans le paragraphe 2.1, ne permet pas de traiter les cas ou il y a plus de 12 signaux produits sur une station ($S_i = 4213597$ pour $n = 12$ cf. paragraphe 2.1). Le cas échéant, la génération de l'ensemble des trames possibles pourra être faite selon la technique proposée dans [6].

4.1 Description

La description algorithmique de l'heuristique "semi-exhaustive" est donnée ci-dessous :

1. sur chaque station i, construction de toutes les partitions possibles de l'ensemble des signaux qui forment l'ensemble F_i^*. Une partition est "possible" si les trames qui la composent ne violent pas la contrainte sur la

taille maximale des données imposée par le protocole de communication et si les échéances des trames sont toutes positives. La période de chaque trame est égale à la plus petite période de production des signaux contenus dans la trame alors que l'échéance est fixé selon l'algorithme donné en Appendice A.

2. sur chaque station i, l'ensemble F_i^* est trié dans l'ordre croissant de consommation de bande passante des trames.
3. $F_{i,1}^*$ est la partition qui minimise la consommation de bande passante sur la station i, les stations sont triées dans l'ordre croissant des $F_{i,1}^*$ (on note e_k l'indice de la $k$ième station sous cet ordre). Pour chaque F_i^*, on décide d'une profondeur p_i au délà de laquelle aucune recherche ne sera faite : les partitions $F_{i,j}^*$ avec $j > p_i$ ne seront pas considérées comme solution possible car peu satisfaisantes en termes de consommation de bande passante. Les valeurs des p_i choisies permettent de contrôler la complexité de l'heuristique. Soit a_i une variable d'indice temporaire initialisée à 1 pour toute station i.
4. la messagerie courante est constituée des partitions $F_{e_1,a_{e_1}}^* \bigcup F_{e_2,a_{e_2}}^* \bigcup ... \bigcup F_{e_m,a_{e_m}}^*$ où m est le nombre de stations du réseau.
5. test de la faisabilité de la messagerie avec l'algorithme d'Audsley :
   (a) la configuration n'est pas faisable : il faut construire la prochaine solution à envisager selon l'algorithme

   ```
   for u :=a_{e_m} downto a_{e_1} do
       if (u<p_u) then a_u :=a_u+1 ; break ;
       else a_u :=1 ; fi
   ```

   Retour à l'étape 4. La figure 2 illustre cette technique de parcours.

   (b) la configuration est faisable : SUCCES.

Dès que la première solution faisable est trouvée, l'algorithme s'arrête. Il serait néanmoins possible de continuer la recherche par exemple jusqu'à ce qu'un nombre fixé de solutions faisables ait été trouvé. Cet algorithme effectue $\prod_{i=1..m} p_i$ appels à l'algorithme d'Audsley et le choix des valeurs de p_i pourra être fait en fonction du temps de calcul disponible. Dans nos expérimentations, les p_i ont toutes été choisies égales à 25 mais nous pensons qu'il est possible d'obtenir de meilleurs résultats en différentiant la valeur de p_i pour chaque station selon des critères à définir. Notons que cet algorithme trouve la solution optimale en termes de bande passante dans le cas particulier où l'ensemble de partitions $F_{e_1,1}^* \bigcup F_{e_2,1}^* \bigcup ... \bigcup F_{e_m,1}^*$ est faisable.

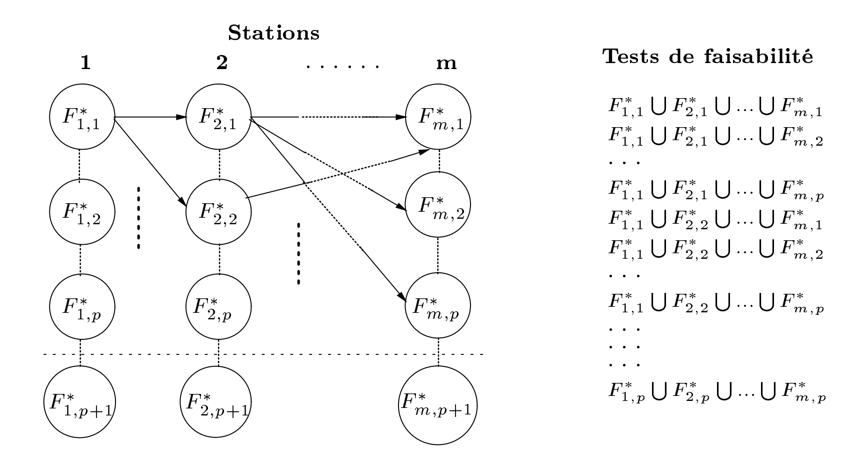


FIG. 2 – Parcours des solutions dans l'heuristique "semi-exhaustive".

4.2 Evaluation de performances

Les expérimentations ont été effectuées avec les paramètres décrits dans le paragraphe 3.2.1 à l'exception du nombre de signaux par stations et du nombre de stations qui ne sont plus aléatoires. En effet, pour être assuré de se trouver dans les conditions d'application de l'heuristique semi-exhaustive (SE), le nombre de signaux est fixé à 10 sur chaque station et le nombre de stations va varier en fonction de la charge utile désirée.

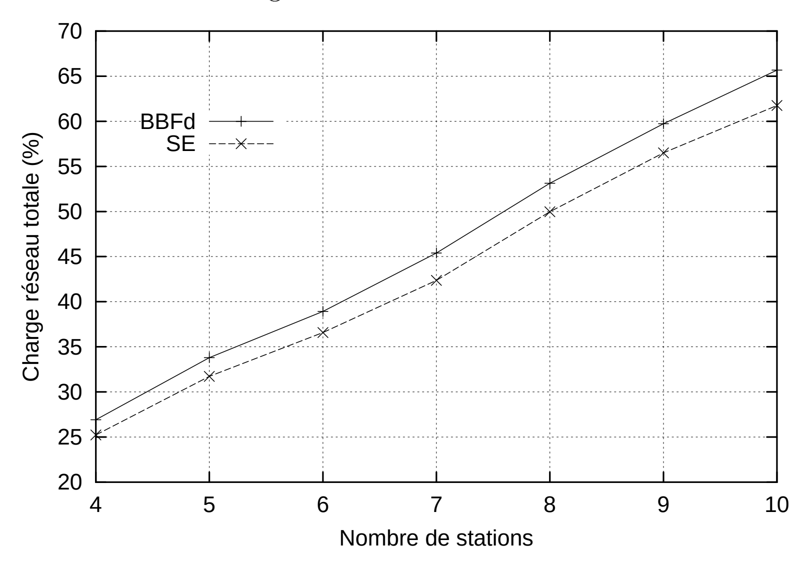


FIG. 3 – Charge réseau totale pour un nombre de stations variant de 4 à 10 avec 10 signaux par stations. La charge utile correspondante est de 11% avec 4 stations, 13,75% avec 5 stations, 16,5% avec 6 stations, 19,25% avec 7 stations, 21,5% avec 8 stations, 24% avec 9 stations et 27,5% avec 10 stations. Résultats moyens sur 100 configurations faisables sous les heuristiques BBFd et semi-exhaustive (SE).

Sur la figure 3, on observe que l'heuristique SE se comporte mieux que BBFd quel que soit le niveau de charge utile. Le gain avec SE se situe entre 5,4 et 6,7% ce qui est significatif en termes de consommation de bande passante. Du point de vue de l'ordonnançabilité, les deux heuristiques ont toutes les deux toujours trouvé des solutions faisables dans nos expérimentations (effectuées avec échéance égale à la période) pour des charges utiles strictement inférieures à 35%. Au delà de ce niveau de charge, des expérimentations préliminaires nous suggèrent que SE est généralement meilleur que BBFd mais le temps de calcul (avec $p_i = 25$) peut alors devenir problématique compte tenu de la raréfaction des solutions faisables.

Conclusion

Dans cette étude, nous avons proposé deux heuristiques pour construire des messageries faisables qui minimisent l'utilisation de la bande passante. Les stratégies proposées sont complémentaires, la première est utilisable pour des problèmes de taille arbitraire, la seconde, plus efficace sur nos tests, ne peut être appliquée que sur des problèmes de petite taille (moins de 12 signaux par ECU). Nos propositions se sont révélées toujours plus efficaces que d'autres stratégies a priori envisageables qui sont best-fit-decreasing, first-fit-decreasing et "un signal par trame".

Les heuristiques peuvent être vraisemblablement améliorées, en particulier en intégrant des stratégies d'optimisation locale partant de la solution trouvée comme cela est fait classiquement dans les problèmes de bin-packing (cf. [5]). Les heuristiques développées pourront servir de point de départ pour d'autres algorithmes d'optimisation en orientant les recherches vers des parties prometteuses de l'espace des solutions. En particulier elles pourront servir pour la création de la population initiale d'un algorithme génétique, population initiale qui influe fortement sur les performances ultérieures de l'algorithme (cf. par exemple [8]).

A Echéance d'une trame après l'ajout d'un signal

La période d'émission d'une trame constituée de plusieurs signaux est la plus petite période de production des signaux la constituant et la transmission de la trame sera synchronisée avec la production du signal de plus petite période. Par contre, l'échéance de la trame n'est pas la plus petite échéance des signaux de la trame à cause de décalages possibles entre les instants de production des signaux et les instants de transmission effectifs. Considérons l'exemple de la figure 4 avec deux signaux s_1, s_2 respectivement de période $T_1 = 10$ et $T_2 = 14$ et d'échéance

relative $D_1 = 10$ et $D_2 = 14$. On observe ainsi que le signal s_2 produit à l'instant 14 n'est effectivement transmis qu'à l'instant 20 et l'échéance relative de cette trame devra être de 8 unités de temps pour respecter la contrainte d'échéance de s_2. On remarque également que les transmissions effectuées aux instants 10 et 40 ne comportent aucune nouvelle valeur de s_2. En pratique, dans une telle situation, la valeur du signal s_2 pourra être retransmise ou non mais il n'y aura aucune contrainte de temps sur la trame induite par le signal s_2. L'échéance de la trame est déterminée de telle façon que pour tous les décalages possibles entre les dates de production des signaux et la transmission de la trame, la contrainte de fraicheur de chacun des signaux soit toujours respectée. Pour déterminer l'échéance, il faut donc trouver le plus grand décalage entre la date de production d'un signal et la transmission de la prochaine trame. Ici, l'échéance de la trame doit être fixée à 6 unités de temps (l'instance de s_2 produite à 42, transmise à 50 et devant arriver avant 56).

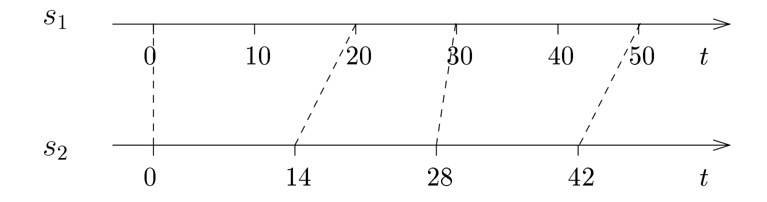


FIG. 4 – Deux signaux de période 10 et 14 constituant une trame dont la transmission est synchronisée sur le signal de période 10. Les flèches pointillées indiquent les dates de transmission du signal de période 14.

On désire insèrer le signal s_i dans la trame f_k déjà constituée des signaux $s_1^k, s_2^k, ..s_n^k$. On note s_{min} le signal de plus petite période de l'ensemble $\{s_1^k, s_2^k, ..s_n^k\} \cup s_i$, la période de f_k devient $T_k^* = T_{min}$. L'échéance relative de f_k est $D_k^* = \min\{D_j - \text{Offset}(T_{min}, T_j) \mid s_j \in \{s_1^k, s_2^k, ..s_n^k\} \cup s_i\}$. La fonction $\text{Offset}(a, b)$ renvoie le plus grand délai possible entre la date de production d'un signal de période $b \geq a$ et la date de transmission de la trame de période a contenant le signal. Sous l'hypothèse de synchronisation des premières dates de mise à disposition, il a été montré dans [10] à l'aide du théorème du reste Chinois généralisé que $\forall k_1, k_2 \in \mathbb{N}$, $k_1 \cdot a - k_2 \cdot b = q \cdot \text{pgcd}(a, b)$ avec $q \in \mathbb{Z}$. Dans notre contexte, on impose $a > k_1 \cdot a - k_2 \cdot b \geq 0$ et donc $\text{Offset}(a, b) = (\frac{a}{pgcd(a,b)} - 1) \cdot pgcd(a, b) = a - pgcd(a, b)$.

Références

[1] M. Abramowitz, I.A. Stegun, *Handbook of Mathematical Functions*, Dover Publications, 1970.

[2] N.C. Audsley, *Optimal Priority Assignment and Feasibility of Static Priority Tasks With Arbitrary Start Times*, Rapport de Recherche YCS164, University of York, 1991.

[3] J. Migge, *rts*, logiciel et manuel d'utilisation disponible à l'adresse `http://www.loria.fr/~nnavet`, 2002.

[4] C. Norström, K. Sandström, M. Ahlmark, F*rame Packing in Real-Time Communication*, Proc. IEEE RTCSA'2000, Korea, version étendue disponible sous forme de Rapport Technique, Mälardalen Real-Time Research Center, 2000.

[5] T. Osogami, H. Okano, *Local Search Algorithms for the Bin Packing Problem and Their Relationships to Various Construction Heuristics*, Proc. IPSJ SIGAL, Sapporo, Japan, 1999, disponible également sous forme d'un Rapport Technique IPSJ numéro 99-AL-69-5.

[6] M. Orlov, *Efficient Generation of Set Partitions*, disponible à l'adresse `http://www.cs.bgu.ac.il/~orlovm/papers/partitions.pdf`, 2002.

[7] E.G. Coffman, M.R. Garey, D.S. Johnson, *Approximation Algorithms for Bin Packing : a Survey*, dans D.S. Hochbaum (ed.), *Approximation Algorithms for NP-Hard Problems.* PWS Publishing Company.

[8] C. H. Westerberg, J. Levine, *Investigation of Different Seeding Strategies in a Genetic Planner*, Proc. EvoWorkshops2001 : EvoCOP, EvoFlight, EvoIASP, EvoLearn, and EvoSTIM, LNCS 2037, Springer-Verlag, 2001.

[9] L. Casparsson, A. Rajnák, K. Tindell, P. Malmberg, *Volcano - a Revolution in On-Board Communications*, Volvo Technology Report 1999.

[10] G. Quan, X. Hu, *Enhanced Fixed-Priority Scheduling with (m,k)-Firm Guarantee*, IEEE Real-Time System Symposium (RTSS), 2000.

[11] K. Tindell, A. Burns, A.J. Wellings, *Calculating Controller Area Network (CAN) Message Response Times*, Control Engineering Practice, vol. 3, number 8, 1995.